Introduction to Angular

By Alex Wilson

What is Angular?

- Angular is a JavaScript-based open-source front-end web application framework
- Mainly maintained by Google and by a community of individuals and corporations
- Addresses many of the challenges encountered in developing single-page applications
- Typescript driven

What is Typescript?

- Typescript is a superset of JavaScript
- Allows for optional static typing, classes, and interfaces
- Very easy for developers that know Java to pick up
- Typescript code is compiled down to pure JavaScript

Angular Core Concepts

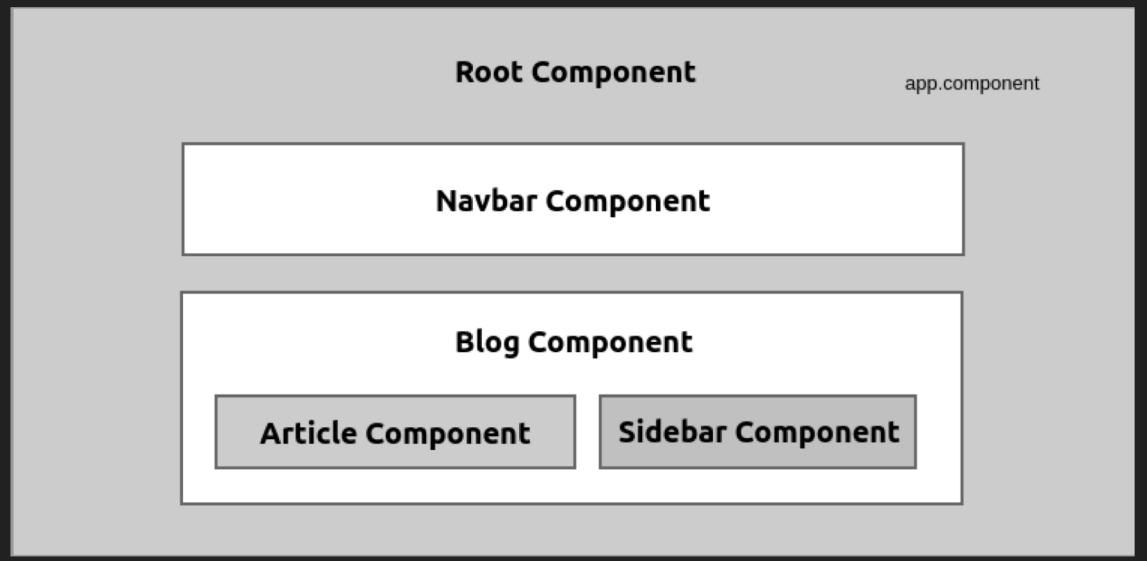
- Module
- Component
- Template
- Service
- Metadata
- Data Binding
- Directive
- Lifecycle hooks

Module

- A module is a collection of services, directives, controllers, filters, and configuration information
- Angular applications use modules as the core mechanism for composition
- Modules export things that other modules can import
- **Keep your modules fine-grained and self-documenting

Component

- A whole application can be modeled as a tree of components
- A component is anything that is visible to the end user and which can be reused many times within an application
- Properties and methods of the component class are available to the template



Components Example

```
import { Component } from '@angular/core';

@Component({
selector: 'my-component',
template: '<div>Hello my name is {{name}}. <button (click)="sayMyName()"> Say my name</button></div>'
})

export class MyComponent {
        name: string;

        constructor() {
                this.name = ‘Jal Irani'
        }
        sayMyName() {
                console.log('My name is', this.name)
        }
}
```

***When we use the <my-component></my-component> selector tag in our HTML, this component will be created, our constructor called, and rendered.

Template

- A template contains HTML used to render part of a view
- Angular leverages native DOM events and properties which dramatically reduces the need for a ton of built in directives

Service

- A service is typically a class with a narrow, well-defined purpose
- Take the burden of commonly used functionality out of the components
- Decorate with @Injectable when we need to inject dependencies into our code

Service Example

- //Example: customer.service.ts
```
import { HttpModule } from '@angular/http'

@Injectable()
export class CustomerService {
  constructor(private http: Http) { }

  getCustomers() {
    return this.http.get('api/customers')
      .map((response: Response) => <Customer[]>response.json().data)
      .catch(this.handleError);
  }

  private handleError(error: Response) {
    console.error(error);
    return Observable.throw(error.json().error || 'Service error');
  }
}
```

Metadata

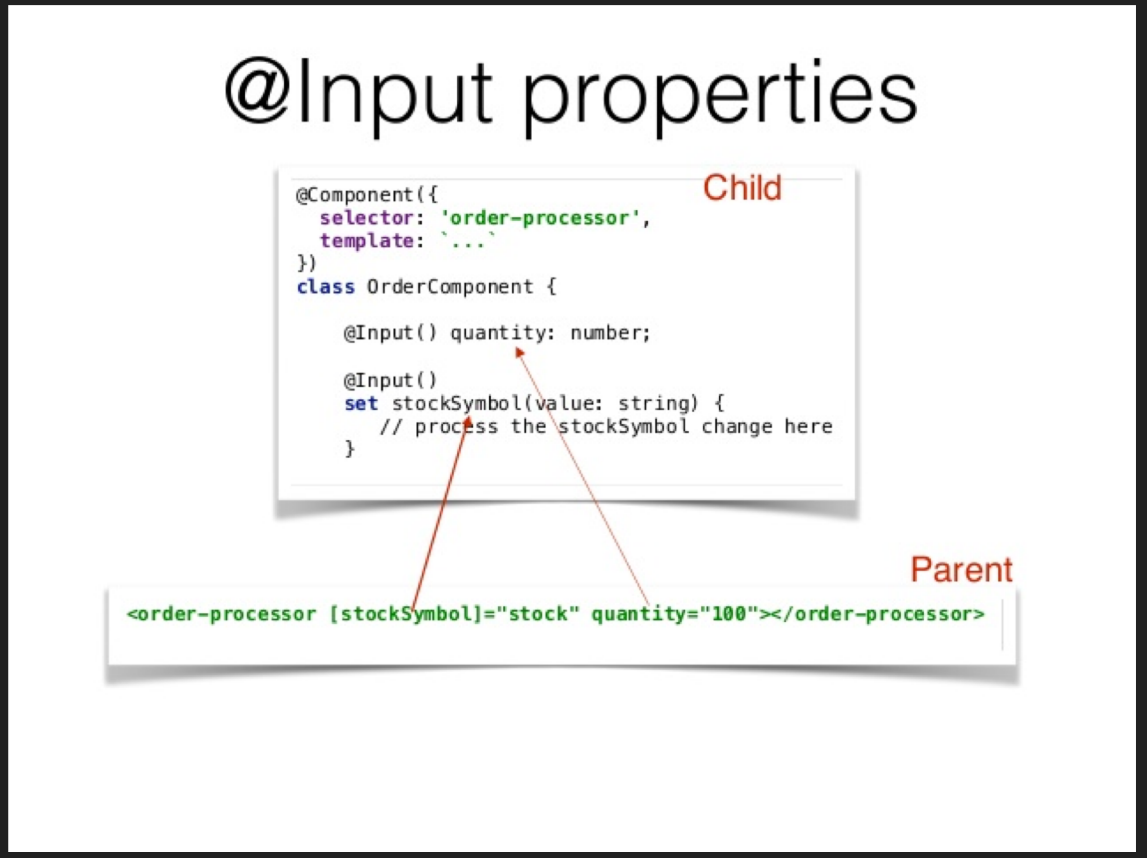
- Metadata allows Angular to process a class
- Annotations that we have already seen:
  - The @Component() annotation tells angular that the class is of type component
  - The @Injectable() annotation tells angular that the class is a service and that it can be injected into another component

Databinding

- Data binding is the process that establishes a connection between the application UI and the typescript functionality
- The template contains angular specific syntax for databinding

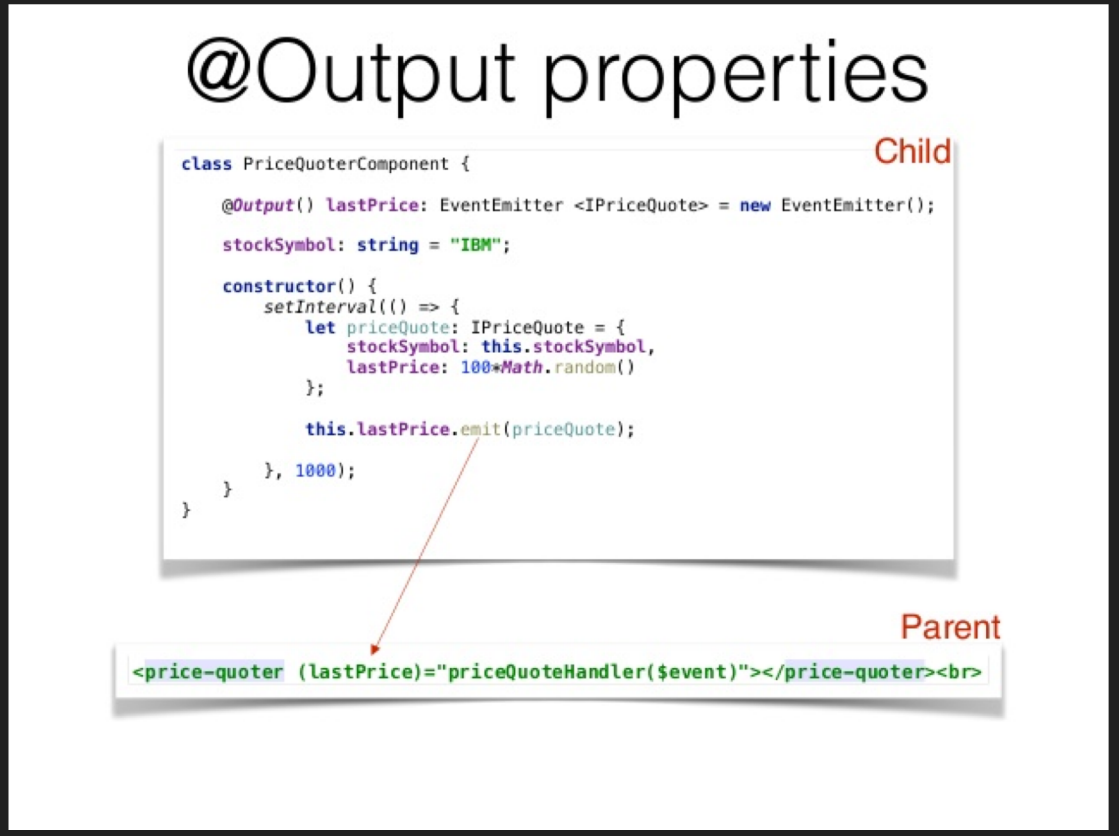
Databinding and Metadata

- Data binding type: One way property binding
  - Used to assign values to properties of another component
- Annotation:
  - The @Input() annotation tells angular that the value of a specified property will get its value from the template.



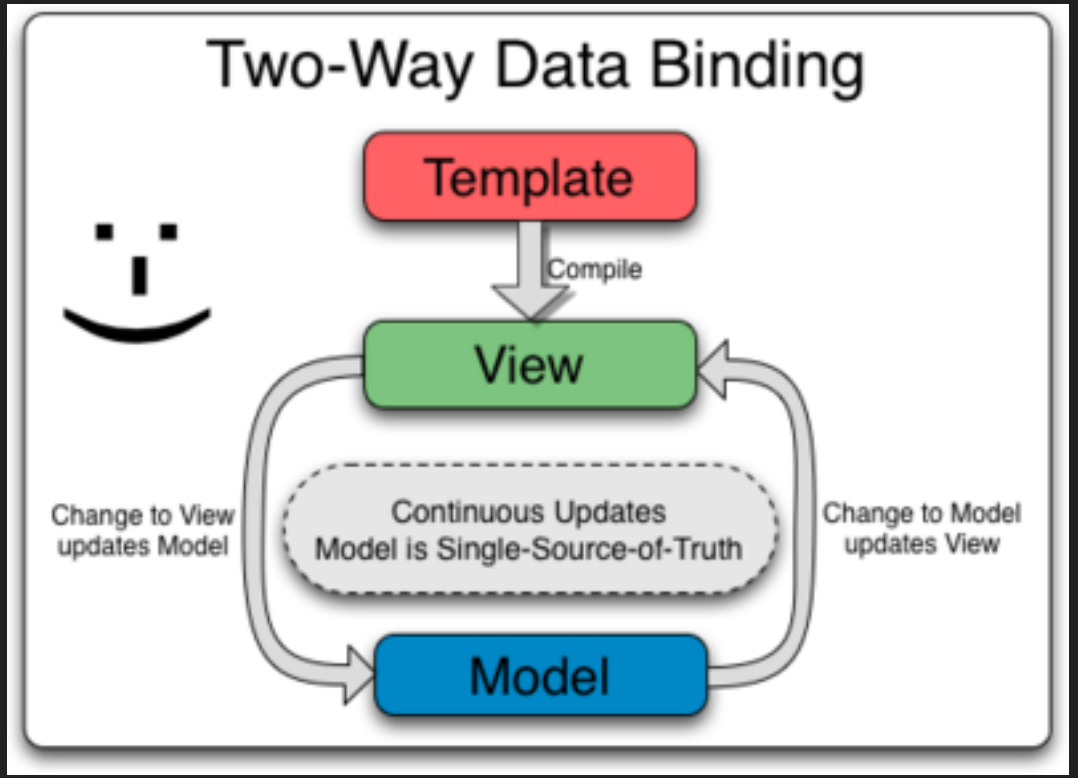
Databinding and Metadata

- Data binding type: One way event binding
  - Used to register callbacks when an event occurs for another component
- Annotation:
  - The @Output() annotation tells angular that a parameter emits a value from the typescript file to the template



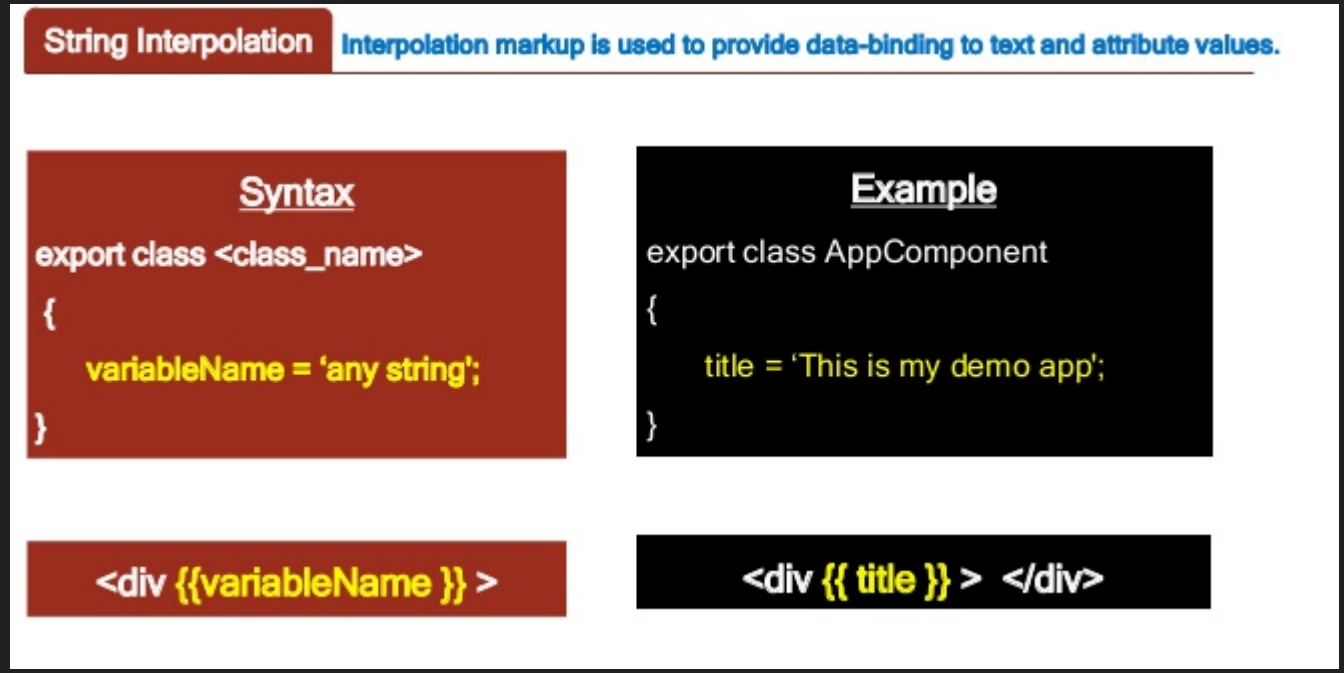
Databinding and Metadata

- Data binding type: two way data binding
  - combines property and event binding into one syntax
- Annotation:
  - Contains an @Output() and a setter so that the value can emit an event but also be set from the template



Data binding

- Data binding type: Interpolation
  - Uses curly brace syntax: {{variable-name}}
  - combines property and event binding into one syntax



Directives

- Directives are instructions in HTML code which will tell Angular how to change/alter/manipulate the code/document when it encounters this directive
- Angular lets us extend HTML by using directives and attributes.
- Types:
  - Structural directive: Changes the DOM layout by removing /adding DOM element
    - ngIf, ngFor, ngSwitch
  - Attribute directive: Changes the appearance or behavior of the element
    - ngStyle, ngClass

Directive Types

- Attribute Directives are called like this, because they are applied like HTML attributes and impact the element they are attached to (e.g. change the styling)
  - Example: <div [ngClass]="{'class-to-be-applied': true}
- Structural Directives are called like this, because they change the structure of the DOM, not only the element on which they sit
  - Examples:
    - <li *ngFor="let item of items">{{item}}
    - <div *ngIf="condition">

Lifecycle hooks

- Commonly use hooks:
  - ngOnChanges() - Respond when Angular (re)sets data-bound input properties
  - ngOnInit() - Initialize the directive/component after Angular first displays the data-bound properties and sets the directive/component's input properties
  - ngAfterViewInit() - Respond after Angular initializes the component's views and child views

- Other hooks:
  - ngDoCheck(), ngAfterContentInit(), ngAfterContentChecked(), ngAfterViewChecked(), ngOnDestroy()

Resources

- Angular Tutorial
  - https://angular.io/docs/js/latest/tutorial/
- Official documentation for Angular
  - https://angular.io/docs/ts/latest/
- Rangle.io
  - https://www.gitbook.com/book/rangle-io/ngcourse2/details